Research Article

Controller Placement in Software-defined Networking Using Silhouette Analysis and Gap Statistic

Sanaz Honarpazhooh [a]* and Amid Khatibi Bardsiri [a]

[a] Computer Engineering Department, Bardsir Branch, Islamic Azad University, Bardsir, Iran.

Article History: Received: 5 April 2021; Accepted: 14 May 2021; Published online: 22 June 2021

Abstract: Software-defined Networks (SDNs) are an emerging network architecture that has been introduced as a way to facilitate network development in such a way that network control is separate from traffic transmission and is directly programmed. Typically, an SDNs relies on a centralized logic control that uses the network's global information to perform operations within the network. Therefore, the SDNs have scalability problems. In this case, the proper placement of the controllers in the SDN increases the quality of service. In this dissertation, a DBSCAN-based clustering method is proposed with the characteristic of non-dependence on determining the number of clusters in advance, which can be used to properly locate the controllers based on the number clusters. The proposed method is simulated and the results show that the proposed method in the worst case load on SDNs works better than similar methods and has improved performance compared to the base method.

Keywords: Controllers Location, Software Defined Network (SDN), Profile Analysis, Statistical Gap, Clustering

1. Introduction

Nowadays, the concept of software-defined networks (SDNs) can be applied in different application areas such as Wireless Sensor Network (WSN), Internet of Things (IoT) communications, cloud–fog computing, Vehicular Ad-hoc Networks (VANETs), etc [1, 2].

The idea of software-defined networks (SDNs) was proposed as a solution for easier network development. In particular, SDN is a model of new networks in which the hardware task is only transport and has not a decision-making task, and this decision-making task is separated from the hardware and performed by the controller [3]. The SDN tries to increase network intelligencing by transferring data control from switch hardware and routers to network virtual software layers and utilizing a centralized software controller, capabilities such as scheduling, scalability, flexibility, and network automation [4]. Literature indicates that SDNs have significant benefits in many ways in comparison to conventional non-SDN networks [5].

Load balancing is a significant concept in SDNs and can significantly contribute to energy savings, which lead to improvement of network utilization of resources. Because SDN can manage the entire network with centralized control, we can distribute traffic more efficiently and easily [6].

The SDNs architecture is based on the separation of the control layer from the data layer in network equipment. It has advantages such as simpler network design, flexibility in service delivery, and better network management. Much research works have been done in this area, assuming a network based on integrated software; But this assumption is not always true for networks in the real environment because to implement the uniform of the SDN architecture requires updating all traditional network equipment [16], but due to the high cost of modifying and updating all traditional network equipment, this is not always possible. Therefore, implementation of this architecture may become impractical for some networks [17]. Therefore, we should try to increase QoS, reduce latency, reduce financial costs and reduce the control messages exchanged in the network, and create a load balance between controllers for making progress in this area.

Reviewing literature, it is seen that load balancing can play a significant role as an efficient method in distributing loads among different routes. Imbalance traffic load in the network can create high latency in some routes and loss of data packets and decreases packet delivery ratio [8, 9].

So far, different load balancing techniques have been proposed that these techniques try to improve the SDN efficency [5, 7]. Reviewing literature, it can be concluded that previous research has been presented in three classes: novel techniques, improved techniques, and reviewing studies [9, 10, 11].

Typically, an SDN network relies on a centralized logic control that uses the network's global information to perform operations within the network. In the SDN network, packet processing control is in accordance with the rules for transferring each packet, which can be installed in switch flow tables. When a new stream first crosses a domain, the controller selects a transmission path for the new stream based on the current state across the network and configures the transmission path setting by setting rules on the associated switch. A simple solution to overcome the above problems is to use multiple distributed controllers that work together to achieve the functionality of a centralized logic controller. This distributed unit can benefit from scalability and reliability. It also has the simplicity of a centralized system [8].

Other challenges include the dependence of the whole network on the controller unit, which if the controller has a problem, the control level may be disturbed, or the control network level may be disrupted. This seems perfectly normal because the controller unit is the core of SDN networks, so solutions must be provided for support and over-fail by other controllers from the control unit so that it does not affect the performance of the entire network in the event of a problem [1, 8].

Selecting the right number of controllers and their location in the network is another challenge of software-based networks that has a significant impact on network productivity. Another problem with software-based networks is the scalability of these networks [2, 6].

So far, many studies have performed to improve the efficiency of techniques in the field of SDNs. Also, literature shows that the SDNs has been considered as a new and practical architecture in the field of SDNs by many researchers in different sciences [12, 15]. On the other hand, many of the previous techniques are provided based on controller placement in this field [12, 13, 14].

Therefore, this paper is proposed a controller placement-based new model for managing some of the challenges mentioned above. In fact, we have tried to offer a novel approach through the selection of the appropriate number of controllers and their location to increase QoS, reduce latency, reduce financial costs, reduce the control messages exchanged on the network, and load balancing between the controllers. This important finding is achieved by locating the controllers correctly. For this purpose, a multipurpose mathematical formula is presented, then it will be mathematically proved that the proposed formula increases the QoS. Also, network traffic will be changed and observations will be recorded based on QoS. It should be noted that in the proposed method, a density-based clustering approach has used to determine the optimal location of the controllers.

The two main goals are considered for this research, which includes scientific and practicalas. We list thes goals as follows:

A) Scientific goals

- Reviewing previous related work
- Optimizing the parameters of increasing QoS, reducing latency, reducing financial costs and reducing control messages exchanged in the network, and creating load balance between controllers
- Comparing the proposed mechanism with the similar mechanism available

B) Practical goals

Software-based networks are the latest revolution in network innovation. All sectors involved in the network industry, including network vendors, network service providers, cloud service providers, and their users, are involved in some way with various aspects of it. Therefore, trying to apply an efficient approach in different application areas that do not take advantage of these networks can be a practical goal of the research.

In the following, we tried to organize the body of the article as follows: In the second part, we covered the background of research and related research. Then we presented the proposed method in the third section and tried to explain the concepts and principles needed to better understand our idea. To evaluate the proposed method and prove the superiority of the efficiency of our approach compared to related methods, experiments were performed in the fourth section, then we compared the results of these tests. In the end, we presented the research results and future directions of the research.

2. Literature Review

Given that the structure of many common networks is hierarchical, the use of such a static architecture will not be sufficient for dynamic communication and the needs of today's companies. Today, one rapidly evolving solution to these challenges is software-defined networks or SDNs. Software-based networking is a new and unique network architecture in which data traffic flow and network control are independent and each is directly programmed [14, 17].

Today, controlling and managing networks limited to hardware is complex and difficult. On the other hand, network tools and equipment are vertically integrated, which makes very poor scalable in the network and the process of network innovation progresses slowly. Therefore, the use of SDNs can enable network infrastructure and virtual machines to define and provide a variety of new services [13]. The SDNs is a new architecture that reduces the complexity of network management by breaking this vertical integration and physically separating the network control level from the level of routers and switches, and centralizing network control to one point in such a way that control of several network devices at one point is performed by software. In fact, SDN has been able to provide intelligent network control with software and integration with changes in the network architecture. Network

administrators can quickly manage, secure, and optimize their organization's network resources using SDN programs that they can write themselves [6]. The SDNs architeture is shown in Figure 1.

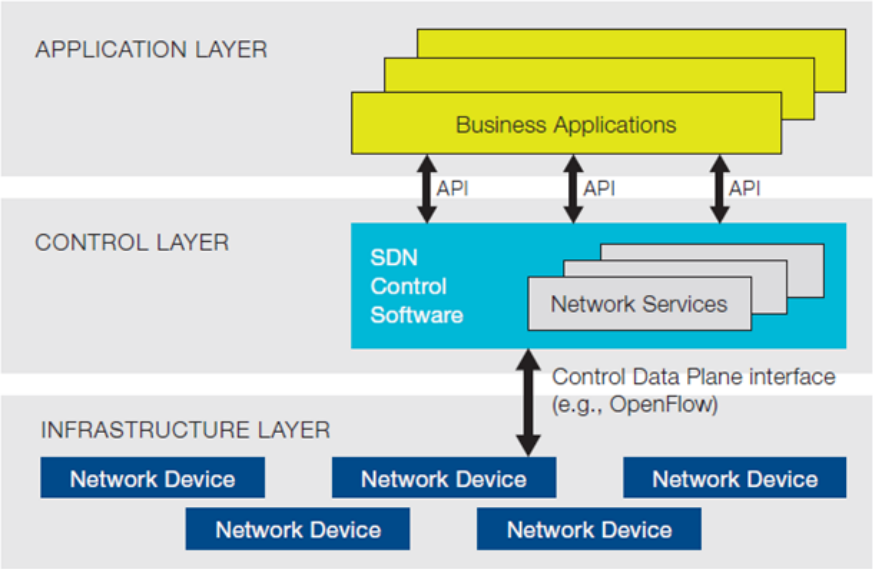

Figure 1. The SDNs architeture [6]

As is seen in Figure 1, network intelligence segment is logically located at the center of the SDN software controllers that maintain the overall network structure; Therefore, the network will look like a logical switch from the point of view of applications. Using SDN, companies and intelligence operators can control and manage the network through a single central controller throughout the network, regardless of the hardware and company.

There are many benefits for using concepts of SDNs that some of them are listed as follows: virtualization technology, centralization, dynamism, optimization, managing power consumption, load balancing, simplification, maximum access, time reduction, cost reduction, etc [18]. Also, there are many applications for SDNs that we try to present several cases as follows [19, 20]:

1) Application in service provider

Industries have recently moved to support both existing networks and SDN models in their network infrastructure. This change in status reduces transfer costs and increases transfer speed.

2) Application in Data Center

Modern datacenters contain thousands of resources (such as processors, memory, and network cards) that are stacked on racks and have thousands of hosts on them. The value of SDN in communication within datacenters is specifically summed up in the ability to virtualize networks and to abstract and automate [21].

3) Application in Slicing network

Sliving is a mechanism for dividing a network infrastructure into several completely separate parts. Each packet forwarding section controls itself without interfering with the other sections, even if they exist on a common infrastructure. the SDN can easily do this with the help of its controllers [7].

4) Application in Traffic Engineering

Traffic engineering is a software method to improve network performance. By creating flexible and programmable networks, SDN makes it easy to control different streams based on customer requirements [22].

So far, a lot of research has been performed on the subject under study. A review of the literature shows two key points. First, by reviewing the research background, it can be concluded that there are a variety and breadth of techniques in this field in different fields of application. Second, there are different challenges such as financial costs, lack of balancing load between controllers, and so on for many techniques due to the variety of requirements and the breadth of new network applications. In this section, we tried to review instances of previous related work in this area. For this purpose, we have prepared Table 1, then we have provided a summary of the main characteristics of some previous instances to increase the readability and ease of understanding of previous ideas.

Table 1. A brief review of the previous research on techniques in the field of SDN

Disadvantage	Advantage	Idea	Year	Method

The problem of generating efficient load balancing, as well as ensuring that packets sent from one client reach the same server	Large groups of clients are sent to the servers without the need for the controller to respond.	Placeing roles on switches	2011	**[23]**
Due to sending all requests to the controller, it creates a lot of load on the controller	This controller consists of three main parts: flow manager, network manager and host manager	Introducing a dedicated controller for high traffic networks called plug-n-server controller	2013	**[24]**
Lacking of attention to SFC error checking Lacking of the study the effects of the proposed method in different dimensions	Investigating the issue of fault tolerance Analyzing errors occured in OpenFlow protocol on SDN	In the proposed method, a node-link error detection architecture and error correction are presented in a way that can be controlled by controllers.	2014	**[25]**

Table 1 (continue). A brief review of the previous research on techniques in the field of SDN

Disadvantage	Advantage	Idea	Year	Method
Not covering all the features in the traditional configuration and configuration of OpenFlow switches	By sending dummy ARP packets over the network, it has been able to modify Layer 2 switch tables in accordance with controller policies	Using the OpenFlow protocol to update traditional Layer 2 switch tables in a hybrid network	2015	**[26]**
The network is not fully visible; the analysis is approximate and highly dependent on external factors	Computing and analyzing the average latency and the worst distance between controllers	Defining the internal latency of controllers	2016	**[6]**
Requires a method to run on hybrid networks	Providing a formula for adding controllers to the SDN network	A multipurpose method for adding controllers to the SDN network	2018	**[27]**
Requires a method to run on hybrid networks	Acceptable performance on different topologies	A mathematical model for locating controllers considering a multipurpose problem	2019	**[28]**
Lacking of reviewing the proposed method in different dimensions of the issue	Performing the intrusion and detection via Suricata and the controller; blocking attempted attacks using Openflow rules automatically	Efficient management of threat detection and secure the protection of a network infrastructure by introducing SDNs	2020	**[17]**
Detecting static flow types developed by IP and port number separation; Lacking of using machine learning algorithms in this field	Proper path selection via TOPSIS in SDN; Reaching the trade-off point and the optimum point in performance metrics with proper path allocation.	Using TOPSIS Decision-Making Algorithm in a SDNs	2021	**[2]**
Lacking iinvestigation of dual-subgradient approach to reduce the time complexity of the proposed TSN	Designing a control policy framework that guarantees transmission time-slot allocations	Application SDN architecture for time sensitive industrial IoT	2021	**[16]**

As shown in Table 1, we try to provide the reader with an overview of the variety of ideas, advantages, and disadvantages of previous work by summarizing the features of recent research.

3. The Proposed Method

In this paper, we propose a novel approach that face a multi-objective problem to increase QoS. For this purpose, first, we have performed silhouette analysis, then a statistical gap and finally a density-based clustering algorithm. The flowchart of the proposed method is shown in Figure 2.

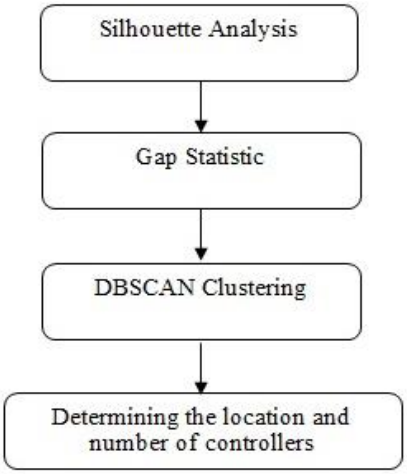

Figure 2. the flowchart of the proposed multipurpose method to increase QoS with optimal location and specified number of controllers

The SDN offers a pattern change in network management and configuration. Due to the network topology, the number of controllers required and their placement location play an important role in providing the specific needs and limitations of the user. Such requirements include delays, failure tolerances, and fair load distribution. These criteria compete with each other, so the best location is not available. Therefore, in the proposed method, we focus on placing the controller in the appropriate place to minimize the release delay and CapEx associated with installing a new controller.

Accordingly, we have used analysis and gap statistics to calculate the optimal number of controllers in a given topology. Also, a density-based clustering method called DBSCAN is used to determine the optimal location of the controllers.

3.1 Silhouette Analysis

Silhouette analysis can play an important role to access data for implementing a controller in a large SDN environment. To manage very large networks, they are usually divided into small segments. Each of these segments will be a cluster under the control of a controller. An important question is how many controllers do a software-based network need to meet the needs of its users? For answering this question, silhouette analysis can be beneficial [29]. This analysis investigates the neighborhood of nodes in a cluster. This criterion can help determine the required number of controllers. The number of controllers in this criterion is discussed based on the minimum number of clusters and the amount of delay. This criterion is computed as follows:

$$\min \sum_{k=1}^{n} L(C_k) \qquad (1)$$

Where, the value of $L(C_k)$ is the delay between clusters. Silhouette analysis algorithm is presented is Figure 3.

Algorithm 1: Silhouette Analysis

1. **Input** $G(V,E,L)$ ←network graph
2. **Input** k ←number of controllers
3. **Input** $d, Clust$ ←distance function handle and clustering algorithm
4. **for** different values of k **do**
   compute intra-cluster variation
5. Calculate the average silhouette coefficient of observations
6. **plot** number of controllers against silhouette coefficients
7. **return** $optimalK$ ←the optimal number of controllers

Figure 3. Silhouette analysis algorithm [18]

3.2 Gap Statistic

The main purpose of this part is considered to determine the exact number of controllers in a way that covers the topology segmentation. For this purpose, the statistical gap makes it possible to compare latency between generating clusters. [41] In fact, it can be said that the main purpose of the gap statistic is to calculate the required number of controllers and maximize the ($Gap_n(k)$) as shown below:

$$Gap_n(k) = E_n^* \{\log(L^*(C_k))\} - \log(L^*(C_k))$$
$$Where\, E_n^* \{\log(L^*(C_k))\} = \left(\frac{1}{B}\right)\sum_b \log(L^*(C_{kb})) \quad (2)$$
$$Gap(k) \geq Gap(k+1) - s_{k+1}$$

In this research, we have used gap statistic algorithm provided in [18] as presented in Figure 4.

3.3 DBSCAN Clustering

In this part, two clustering algorithms are presented: Partition Around Medoids (PAM) and DBSCAN. The first algorithm, PAM is used as basic algorithm when evaluating the efficiency of the proposed method. The second algorithm is used in third step of the proposed method.

The PAM clustering algorithm is one of the most common types of k-medoids algorithms, which converges to the local optimal at the beginning of the algorithm due to the random selection of representative and non-representative objects, and does not necessarily produce the optimal answer in clustering [18]. This algorithm is object-based and selects the representative of the clusters from among the data itself, rather than averaging them. In fact, the medoid of a cluster is the most central element of a cluster. The purpose of this method is to reduce the sensitivity to large amounts in the data set.

Algorithm 2: Gap Statistic

1. **Input** $G(V,E,L)$, k ← network graph, number of controllers
2. **Input** $d, Clust$ ← distance function handle and clustering algorithm
3. **for** varying k **do**
4. Run clustering algorithm on original data to find k clusters
5. Calculate $L(C_k)$ ← intra-cluster latency variation
6. Generate a set of reference datasets same ← size as original data
7. **for** $b=1,2,.....B$ **do**
8. Calculate $L^*(C_{kb})$ ← intra-cluster latency variation of reference
9. Calculate the mean of reference datasets ← as per equation (5)
10. Compute $Gap_n(k)$ ←gap value
11. Calculate s_k ← standard deviation of B Monte Carlo replicates
12. Take the gap value that satisfy equation (6)
13. **return** $optimalK$ ← optimal number of controllers

Figure 4. Gap statistic algorithm [18]

The DBSCAN algorithm is a very common method in the family of density-based clustering algorithms for specific data. The most important features of this algorithm include the ability to detect clusters with arbitrary shapes, the ability to cluster data with noise, and low time complexity. In this clustering method, each data belonging to cluster C is available to the density of other data belonging to that cluster and no other data is available to the density [30]. The steps of this algorithm are listed below:

- Selecting the desired point *X*.
- We find all the points that are available in density from *X*. If *X* is a core point, we form a cluster, and if *X* is a boundary point, no point will be available from *X* on density.
- Repeating the above steps for the next points.

The most important advantages of this algorithm are presented in the following:

- No need for previous information such as the number of clusters.
- Requires two parameters of neighborhood radius (ε) and the minimum number of available objects and is not sensitive to the order of points in the database.
- Supports the concept of noise.
- This algorithm can not cluster data sets with high-density differences, because the values of the minimum number of objects and the neighborhood radius can not be suitable for all clusters.
- This algorithm has a high computational overhead. For example, in dense areas, neighborhoods have a lot in common. As a result, it takes into account many iterative points in its calculations, therefore, reduces the efficiency of this algorithm.

In the following, the output of the DBSCAN clustering algorithm is an instance of the data in Figure 5.

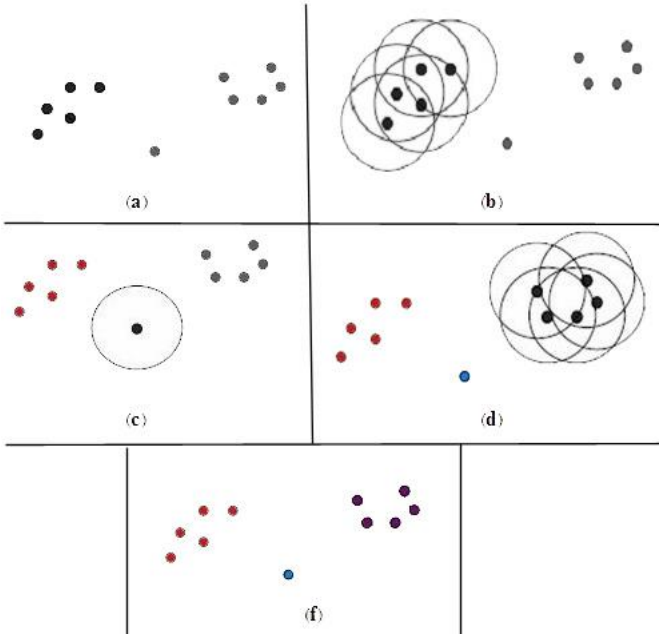

Figure 5. The output of implementing the DBSCAN algorithm [30]

As can be seen, this algorithm has good clustering the data.

3.4 Determining the location and number of controllers

In this research, different methods of determining the optimal number of controllers are investigated for deployment in a given network, with a special focus on the purpose of delay. Although the issue of controller placement has been studied in the past, there are no studies on solutions to determine the optimal number of controllers to deploy for a particular topology because the number of controllers was predetermined in previous studies. However, the optimization solution presented in this paper uses the DBSCAN clustering algorithm, which is density-based and does not require initial quantification, and solves the problem of the need to determine the number of controllers in advance. Most previous studies have used innovative algorithms that typically have the challenge of slow execution speeds. However, the proposed algorithm based on DBSCAN clustering optimizes accuracy rather than speed, which we believe is more important in placing the controller.

4. Experimental Results and Discussion

In this section, the proposed method is implemented first, then the implementation results are presented. Finally, the efficiency of the proposed method is compared with other related methods and evaluation results are reported. Hence, we have organized this section into four sub-sections.

4.1. Data set

The map used as the input to the problem is shown in Figure 6. The methods are implemented on the *Internet 2 OS3E database*, which can be accessed from a reputable *Internet Topology Zoo*.

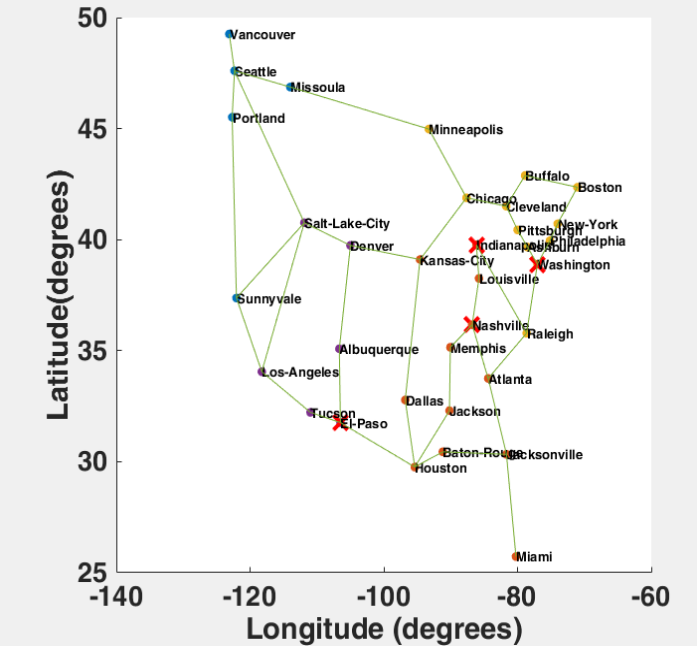

Figure 6. Test environment

4.2. Implementation Tool

In this research, the proposed method has been implemented using MATLAB simulator, 2015 version. In the following, we have presented the specifications of the system in which the proposed method was implemented.

Table 2. System specifications for simulation and evaluation of results

Specifications	Hardware/Software
Win 7	Operating system
64 bit	Type of operating system
8 GB	RAM memory
Intel, 7 core; Q 720 @ 1.60GHz 1.60 GHz	cpu

4.3. Implementation Results

In this section, the gap value is first calculated for the proposed method based on DBSCAN and the PAM method, then the results are shown in Figures 7 and 8.

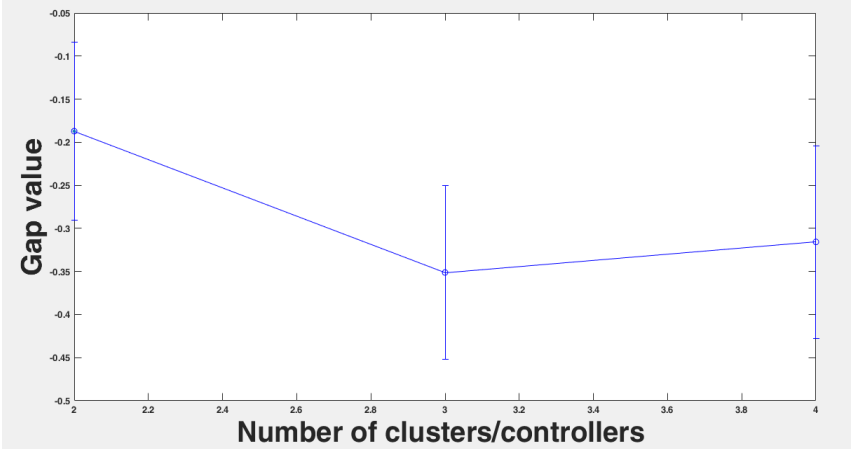

Figure 7. Gap value for PAM

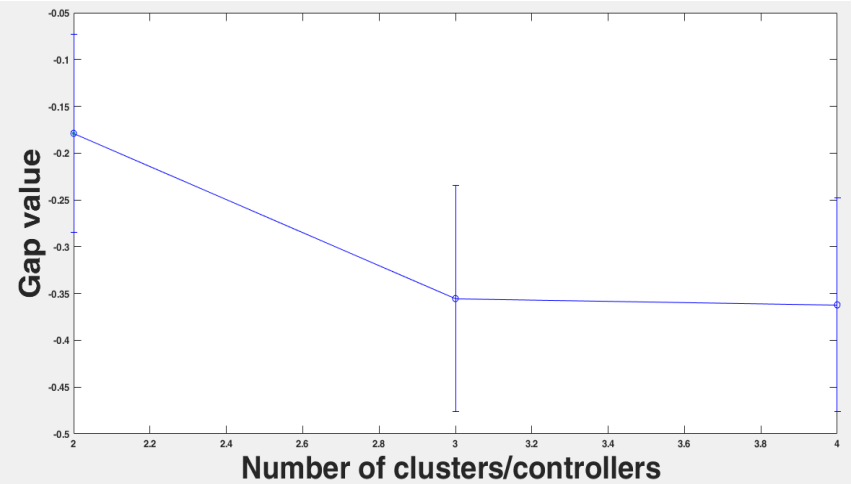

Figure 8. Gap value for the proposed method based on DBSCAN

As is shown in Figures 7 and 8, the amount of gap is provided almost less by the proposed method for 4 cluster in comparison PAM method. In the next experiment, silhouette analysis are performed on controllers for PAM and our method then results are depicted in Figures 9 and 10.

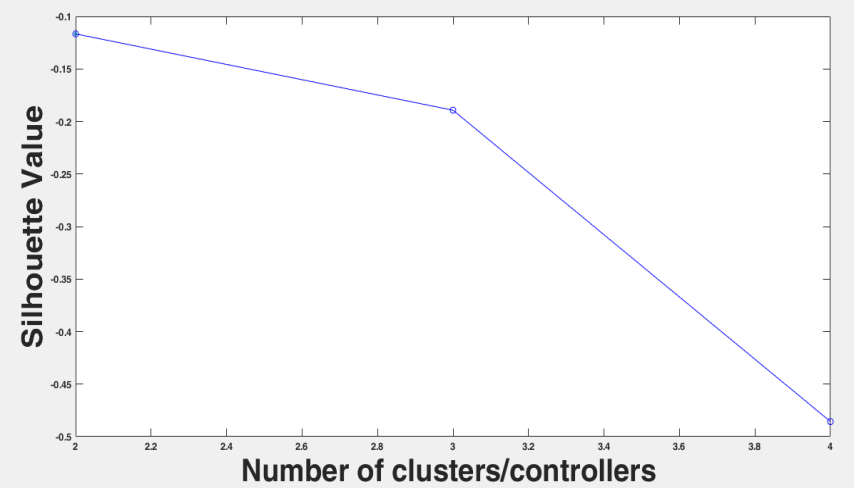

Figure 9. Silhouette analysis on controllers for PAM

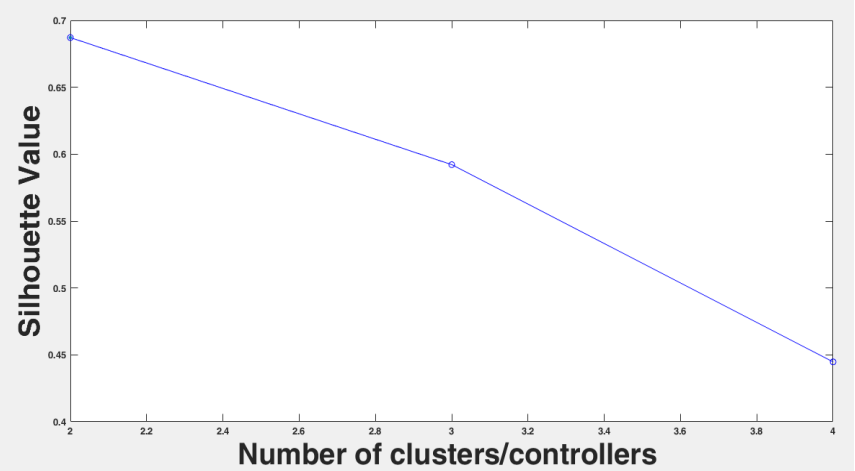

Figure 10. Silhouette analysis on controllers for the proposed method

As seen in Figures 9 and 10, the minimum amount of silhouette analysis is provided by the proposed method for the four clusters. In fact, it can be said that the proposed method has presented better results in comparison with the PAM algorithm.

In this part, the silhouette analysis is performed on clusters for PAM and our method then results of this analysis is shown in Figures 11 and 12.

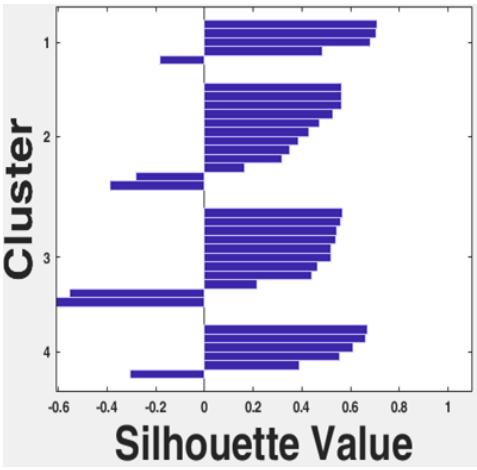

Figure 11. Silhouette analysis on clusters for PAM

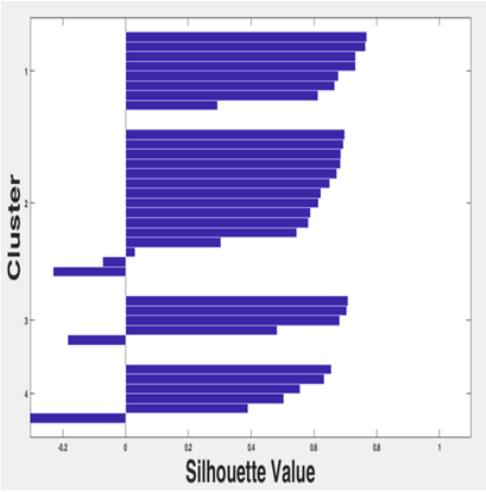

Figure 12. Silhouette analysis on clusters for our method

In the following, the final location of the four clusters with different colors and the controllers with indigo color is shown in Figure 13. Also, the identified clusters for the four clusters and their centers are shown in Figure 14.

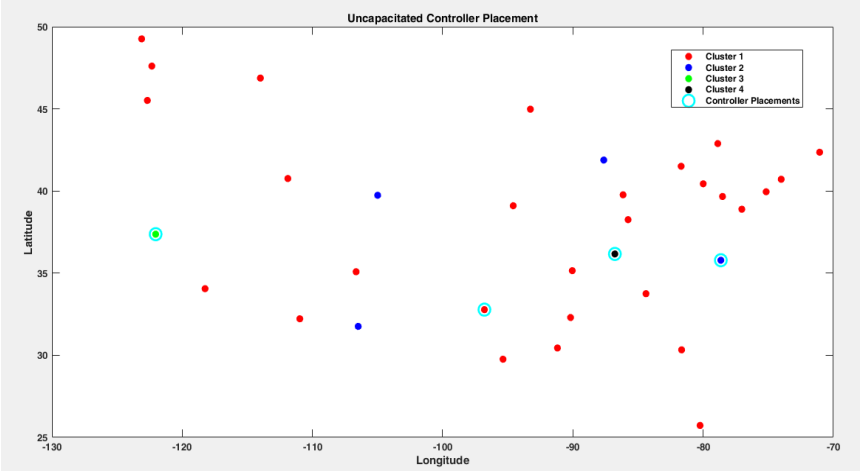

Figure 13. Clusters and controllers location

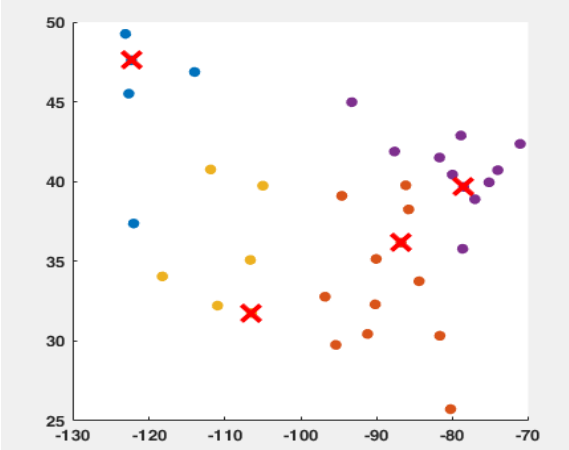

Figure 14. Detected clusters and centers for 4 clusters

In this section, we tried to identify each cluster individually for better identification of clusters. Therefore, the identified clusters and centers are depicted for four clusters in Figure 15.

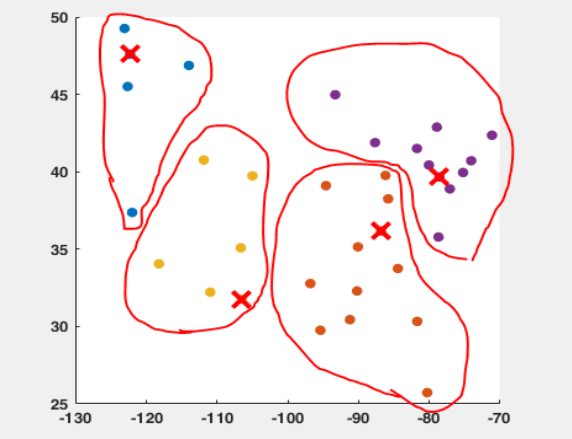

Figure 15. The identified clusters and centers for four cluster

In this section, the efficiency of the proposed method and the PAM method are compared based on the amount of latency in the worst case and the average case, then the evaluation results are shown in Figures 16, 17, and 18, respectively.

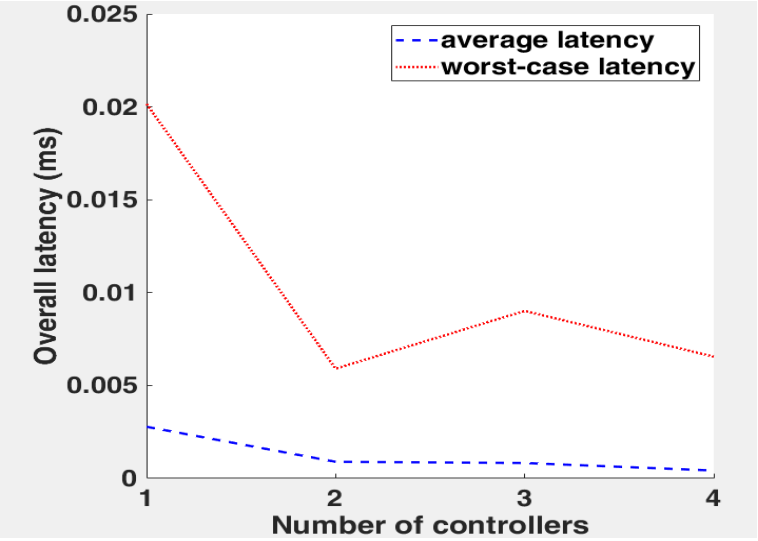

Figure 16. The latency rate

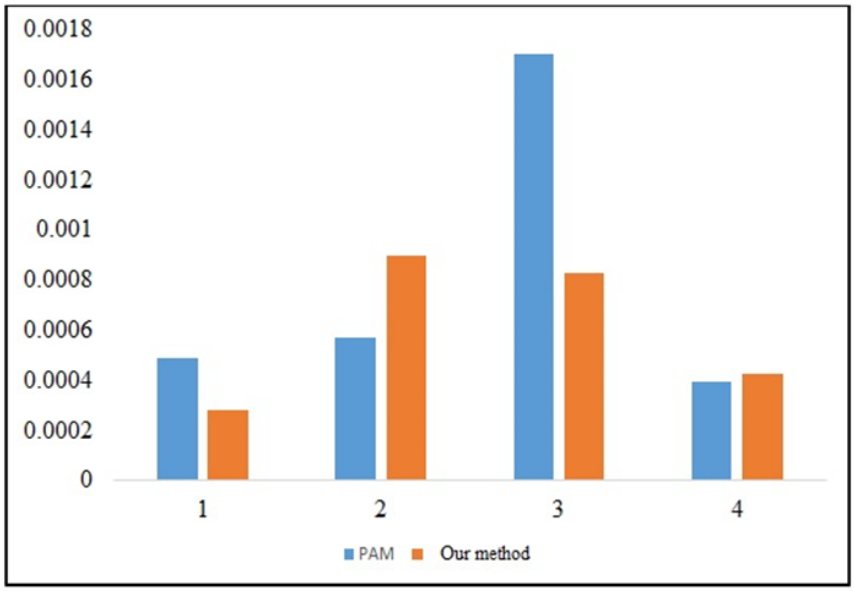

Figure 17. The latency rate obtained for the PAM algorithm and the proposed algorithm in the worst case latency

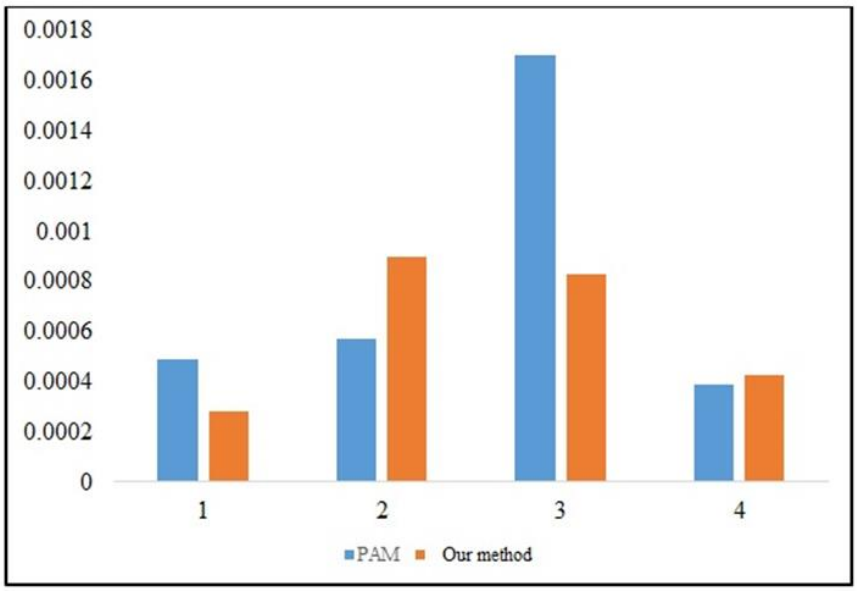

Figure 18. The latency rate obtained for the PAM algorithm and the proposed algorithm in the average case latency

As shown in Figures 17 and 18, the lowest latency rate is presented by the proposed method in the worst and worst case. Also, the results of Figures 17 and 18 show that the highest latency rate is reported for Algorithm PAM. In this part, the rate of delay created by the proposed method and the PAM method is evaluated by changing the number of clusters, then the evaluation results are reported in Table 3.

Table 3. Evaluating the latency rate when changing cluster count for PAM and our method

The proposed method		PAM method		Cluster count
the average case latency	the worst case latency	the average case latency	the worst case latency	
0.0093	0.0247	0.0076	0.0159	Average of one cluster
0.0020	0.0083	0.0020	0.0098	Average of two clusters
0.0020	0.0123	0.00096	0.0067	Average of three clusters
0.0011	0.0089	0.00043	0.0053	Average of five clusters
0.0012	0.0118	0.00028	0.0041	Average of six clusters

As is seen in Figures 17, 18, and Table 3, evaluation results show that there is not much difference between the latencies created for 4 and 5 clusters when using the proposed method. Therefore, it seems logical to consider 4 due to the reduction in the cost of the number of clusters. The clustering output is illustrated by changing the number of clusters when applying the proposed method in order to better show and understand the superiority of the results created by the proposed method in Figures 19 -22.

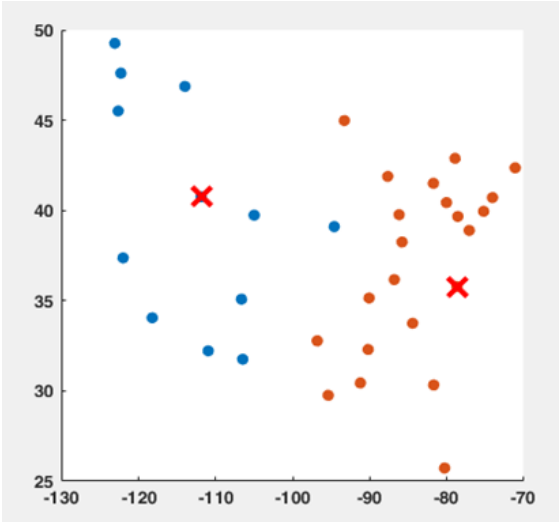

Figure 19. Detected clusters and centers for 2 clusters

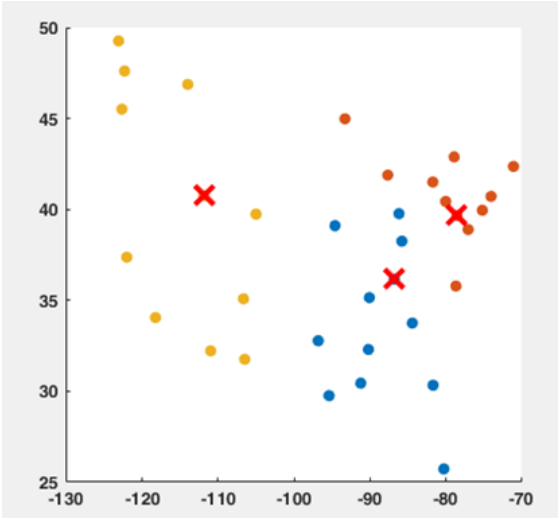

Figure 20. Detected clusters and centers for 3 clusters

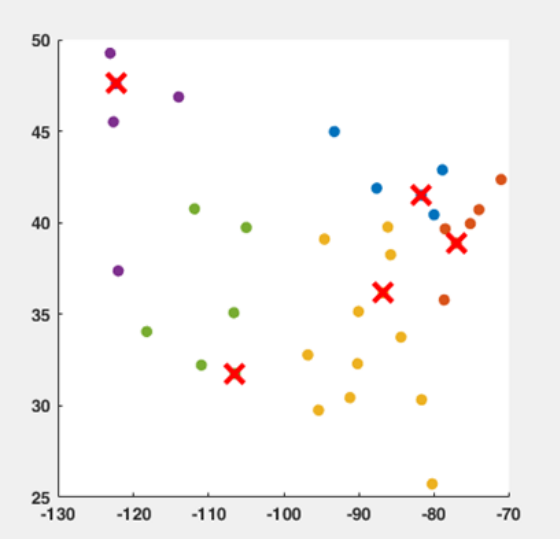

Figure 21. Detected clusters and centers for 5 clusters

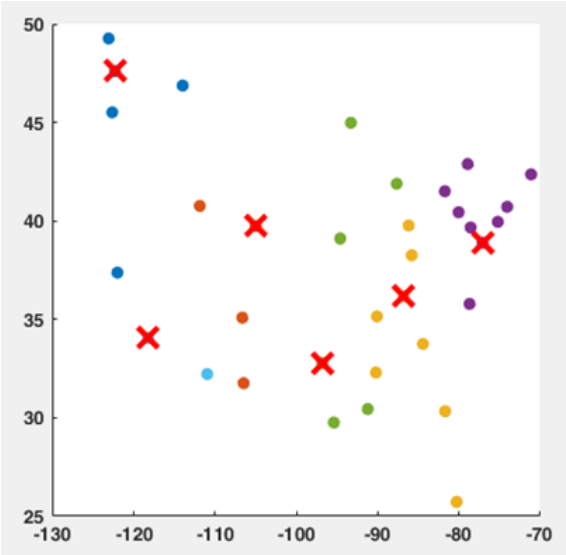

Figure 22. Detected clusters and centers for 6 clusters

4.4 Discussion

In this section, a comparison is made between the proposed method and the PAM method due to the latency created by the methods. For this purpose, the amount of latency created by the methods is computed by changing the number of clusters in the worst and average cases, then the results are reported in Tables 4 and 5.

Table 4. Comparing the proposed method and the PAM method in terms of latency rate in the worst case

Method	First cluster	Second cluster	Third cluster	Fourth cluster
PAM	0.0097	0.0060	0.0116	0.0054
The proposed method	0.0092	0.0059	0.0090	0.0066

Table 5. Comparing the proposed method and the PAM method in terms of latency rate in the average case

Method	First cluster	Second cluster	Third cluster	Fourth cluster
PAM	0.0004896	0.0005720	0.0017000	0.0003910
The proposed method	0.0002800	0.0008957	0.0008296	0.0004252

Evaluation of the experimental results shows that there are similarities and differences between the performance of the PAM algorithm and the proposed method in terms of the latency rate created by these methods, in the worst case. As a function similarity, it can be said that there is little difference between the latency rates created by the tested methods.

On the other hand, by comparing the results of Tables 4 and 5, it can be concluded that there is the main difference in latency rate provided by these two algorithms in the worst and average cases. In other words, the latency rate created by the proposed method is lower compared to the PAM method in the worst case. As can be seen in the results of Table 5, the proposed method has performed better in all cases except the fourth cluster in the worst case of load distribution on SDN s.

Therefore, it can be concluded that the proposed method provides a the lowest latency rate in the worst case of load distribution on the network and presents better Comparison of the results of Table 5 shows that the PAM algorithm has provided better performance in the average mode for all clusters except the first cluster compared to the proposed method. Also, observing the test results in Table 5, it can be concluded that there is not much difference between the performance of these two methods in the average state.

On the other hand, the comparison of the results of Tables 4 and 5 empirically proves that the proposed method has performed better than the PAM method in high network load conditions. This achievement can show that the proposed method can be used as an efficient approach in this field. performance compared to the PAM method.

Comparing the results of Tables 4 and 5, it can be concluded that the experimental results did not show much difference between the results of the fourth and fifth clusters. So it seems that cluster number considered 4 because it imposes less cost on the system.

5. Conclusion and Future Works

In recent years, the idea of a software-defined network has become dominant and has been successfully applied in several network environments. Open Flow software is open-source software for implementing network architecture defined by software and is a potential approach to provide abstract elements for reconstruction or segmentation of network topologies. The main purpose of using Open Flow is to increase the flexibility of network management and traffic routing.

Also, its programmable control environment can help to quickly and easily implement various policies regarding network tools for flight configuration instead of redundant manual configuration.

Accordingly, we hope to be able for using Open Flow to manage and deploy applications that provide our needs to improve network transmission performance, increase packet transfer flexibility, simplify network complexity, and manage the ability to define the network. The energy consumed by network devices such as switches is not suitable for using its links. In other words, energy consumption is high compared to the amount of traffic carried on the network.

In general, the energy efficiency of the Internet is very low. In general, turning on a switch consumes the most power, and using an inactive link to full state consumes only 8% of the extra power. Although all nodes are active all the time, it has been shown that up to 40% of the link capacity is used. Therefore, we can turn off unnecessary links and switches in the network to save energy.

With a global network view, the SDN controller is able to determine the power status and routing decisions on the network. Additionally, the SDN controller can encapsulate the stream table inputs and label each packet to display routing information.

In this study, we first reviewed the literature on the SDNs with a focus on aspects of SDNs features, structure, and application. After introducing the controller, we looked at the history of similar research in this field. We should say that a review of the subject literature has shown that there is a major challenge of scalability in this area. Therefore, we tried to take a step towards solving the current challenge by presenting a DBSCAN-based clustering approach for controller replacement.

For this purpose, the paper tried that describe concepts related to silhouette analysis and statistical gap in this field. In fact, the proposed approach focuses on controller placement to minimize release latency and CapEx associated with installing a new controller. The evaluation results show that the proposed method has a better performance compared to other compared methods such as PAM.

For future work on this research, the following can be considered:

- Using combined K-means algorithms to achieve better results
- Implementing the algorithm in a wider environment
- Using other databases in implementation.

References

[1] R. Amin, I. Pali and V. Sureshkumar, "Software-Defined Network enabled Vehicle to Vehicle secured data transmission protocol in VANETs," Journal of Information Security and Applications, Vol. 58, pp.102729, 2021.

[2] A. Shirmarz and A. Ghaffari, "Automatic Software Defined Network (SDN) Performance Management Using TOPSIS Decision-Making Algorithm," Journal of Grid Computing, vol. 19, no. 2, pp.1-21, 2021.

[3] M. Yu, A. Wundsam and M. Raju, "NOSIX: A lightweight portability layer for the sdn os, SIGCOMM Comput. Commun. Rev.," vol. 44, no. 2, pp. 28–35, 2014.

[4] Y. Kanaumi, S. Saito, E. Kawai, "Toward large-scale programmable networks: Lessons learned through the operation and management of a wide-area Open Flow-based network," Network and Service Management (CNSM), 2010.

[5] S. Khorsandroo, A.G. Sánchez, A.S. Tosun, J.M.A. Rodríguez and R. Doriguzzi-Corin, "Hybrid SDN evolution: A comprehensive survey of the state-of-the-art. Computer Networks," p.107981, 2021.

[6] Shi. Weisong, et al. "Edge computing: Vision and challenges." IEEE Internet of Things Journal, vol. 3, pp. 637-646, 2016.

[7] M. Qilin and S. Weikang, "A load balancing method based on SDN," in Measuring Technology and Mechatronics Automation (ICMTMA), Seventh International Conference on, pp. 18-21, 2015.

[8] M. Ider, and B. Barekatain, "An enhanced AHP–TOPSIS-based load balancing algorithm for switch migration in software-defined networks," The Journal of Supercomputing, vol. 77, no. 1, pp.563-596, 2021.

[9] B. Pourghebleh, and V. Hayyolalam, "A comprehensive and systematic review of the load balancing mechanisms in the Internet of Things, " Cluster Computing, pp.1-21, 2019.

[10] A.A. Neghabi, N.J. Navimipour, M. Hosseinzadeh, and A. Rezaee, "Load balancing mechanisms in the software defined networks: a systematic and comprehensive review of the literature, ". IEEE Access, vol. 6, pp.14159-14178, 2018.

[11] E.J. Ghomi, A.M. Rahmani, and N.N. Qader, "Load-balancing algorithms in cloud computing: A survey, " Journal of Network and Computer Applications, vol. 88, pp.50-71, 2017.

[12] B.P.R. Killi, and S.V. Rao, "Controller placement in software defined networks: A comprehensive survey, " Computer Networks, vol. 163, p.106883, 2019.

[13] J. Xie, F.R. Yu, T. Huang, R. Xie, J. Liu, C. Wang and Y. Liu, "A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges, " IEEE Communications Surveys & Tutorials, vol. 21, no. 1, pp.393-430, 2018.

[14] J.H. Cox, J. Chung, S. Donovan, J. Ivey, R.J. Clark, G. Riley, and H.L. Owen, " Advancing software-defined networks: A survey, ". IEEE Access, vol. 5, pp.25487-25526, 2017.

[15] [15] Y.E. Oktian, S. Lee, H. Lee, and J. Lam, "Distributed SDN controller system: A survey on design choice. computer networks, " vol. 121, pp.100-111, 2017.

[16] V. Balasubramanian, M. Aloqaily, and M. Reisslein, M., "An SDN architecture for time sensitive industrial IoT, ". Computer Networks, vol. 186, p.107739, 2021.

[17] O. Flauzac, E.G. Robledo, C. Gonzalez, F. Mauhourat and F. Nolot, "SDN Architecture to prevent attacks with OpenFlow," In 8th International Conference on Wireless Networks and Mobile Communications (WINCOM) pp. 1-6, IEEE, 2020.

[18] L. Mamushiane, M. Joyce, and A.L. Albert, "Given a SDN Topology, How Many Controllers are needed and Where Should They Go?, " IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN). IEEE, PP. 1-5, 2018.

[19] F. Da Rocha, C. Paulo, and M. Edjard Souza, "A survey on fault management in software-defined networks, " IEEE Communications Surveys & Tutorials, vol. 19, no. 4, pp. 2284-2321, 2017.

[20] A. Mayoral, et al., "SDN orchestration architectures and their integration with cloud computing applications, " Optical Switching and Networking, vol. 26, pp. 2-13, 2017.

[21] S. Abdi, V. Hajizadeh, S. Rowshanrad, M. Namvarasl, and M. Keshtgary, "A survey on SDN, the future of networking, " Journal of Advanced Computer Science & Technology. Vol 3, no. 2, PP. 232-248, 2014.

[22] R. Mohammadi, R. Javidan, M. Keshtgari, and N. Rikhtegar, "SMOTE: An Intelligent SDN-Based Multi-Objective Traffic Engineering Technique for Telesurgery," IETE Journal of Research, pp.1-11, 2021.

[23] R. Wang, D. Butnariu, and J. Rexford, "OpenFlow-Based Server Load Balancing Gone Wild," Hot ICE, vol. 11, 12-12, 2011.

[24] D. Hock, "Pareto-optimal resilient controller placement in SDN-based core networks," 1–9, 2013.

[25] J.M.S Vilchez, G. B. Y. Imen, and C. Noël, "Self-healing mechanisms for software defined networks," 8th International Conference on Autonomous Infrastructure, Management and Security (AIMS 2014)., 2014.

[26] C. Jin, C. Lumezanu, Q. Xu, Z.L. Zhang, and G. Jiang, "Telekinesis: Controlling legacy switch routing with openflow in hybrid networks, " In Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research, pp. 1-7, 2015.

[27] T. Das, and M. Gurusamy, "INCEPT: INcremental ControllEr PlacemenT in software defined networks," In 27th International Conference on Computer Communication and Networks (ICCCN), pp. 1-6, 2018.

[28] S. Mohanty, P. Priyadarshini, B. Sahoo, and S. Sethi, "A Reliable Capacitated Controller Placement in Software Defined Networks," In 3rd International Conference on Computing Methodologies and Communication (ICCMC), pp. 822-827, IEEE, 2019.

[29] Chapman and Hall, "Data Clustering: Algorithms and Applications," CRC Press, 2006.

[30] M. Li, X. Bi, L. Wang, and X. Han, "A method of two-stage clustering learning based on improved DBSCAN and density peak algorithm, " Computer Communications, vol. 167, pp.75-84, 2021.